
Subject: [RFC/PATCH 3/8]: CGroup Files: Move the release_agent file to use typed handlers

Posted by [Paul Menage](#) on Tue, 13 May 2008 06:37:10 GMT

[View Forum Message](#) <> [Reply to Message](#)

Adds cgroup_release_agent_write() and cgroup_release_agent_show() methods to handle writing/reading the path to a cgroup hierarchy's release agent. As a result, cgroup_common_file_read() is now unnecessary.

As part of the change, a previously-tolerated race in cgroup_release_agent() is avoided by copying the current release_agent_path prior to calling call_usermode_helper().

Signed-off-by: Paul Menage <menage@google.com>

kernel/cgroup.c | 103 ++++++-----
1 file changed, 35 insertions(+), 68 deletions(-)

Index: cgroup-2.6.25-mm1/kernel/cgroup.c

=====

--- cgroup-2.6.25-mm1.orig/kernel/cgroup.c

+++ cgroup-2.6.25-mm1/kernel/cgroup.c

@@ -89,11 +89,7 @@ struct cgroupfs_root {

/* Hierarchy-specific flags */
unsigned long flags;

- /* The path to use for release notifications. No locking
- * between setting and use - so if userspace updates this
- * while child cgroups exist, you could miss a
- * notification. We ensure that it's always a valid
- * NUL-terminated string */

+ /* The path to use for release notifications. */
char release_agent_path[PATH_MAX];
};

@@ -1371,6 +1367,22 @@ static void cgroup_file_unlock(struct cg
mutex_unlock(&cgroup_mutex);
}

+static int cgroup_release_agent_write(struct cgroup *cgrp, struct cftype *cft,
+ char *buffer)
+{
+ BUILD_BUG_ON(sizeof(cgrp->root->release_agent_path) < PATH_MAX);
+ strcpy(cgrp->root->release_agent_path, buffer);
+ return 0;
+}
+

```

+static int cgroup_release_agent_show(struct cgroup *cgrp, struct cftype *cft,
+    struct seq_file *seq)
+{
+    seq_puts(seq, cgrp->root->release_agent_path);
+    seq_putc(seq, '\n');
+    return 0;
+}
+
static ssize_t cgroup_write_X64(struct cgroup *cgrp, struct cftype *cft,
    const char *buffer)
{
@@ -1435,10 +1447,6 @@ static ssize_t cgroup_common_file_write(
    else
        clear_bit(CGRP_NOTIFY_ON_RELEASE, &cgrp->flags);
    break;
- case FILE_RELEASE_AGENT:
-     BUILD_BUG_ON(sizeof(cgrp->root->release_agent_path) < PATH_MAX);
-     strcpy(cgrp->root->release_agent_path, buffer);
-     break;
default:
    retval = -EINVAL;
    goto out2;
@@ -1533,49 +1541,6 @@ static ssize_t cgroup_read_X64(struct cg
    return retval;
}

-static ssize_t cgroup_common_file_read(struct cgroup *cgrp,
-    struct cftype *cft,
-    struct file *file,
-    char __user *buf,
-    size_t nbytes, loff_t *ppos)
-{
-    enum cgroup_filetype type = cft->private;
-    char *page;
-    ssize_t retval = 0;
-    char *s;
-
-    if (!(page = (char *)__get_free_page(GFP_KERNEL)))
-        return -ENOMEM;
-
-    s = page;
-
-    switch (type) {
-    case FILE_RELEASE_AGENT:
-    {
-        struct cgroupfs_root *root;
-        size_t n;
-        mutex_lock(&cgroup_mutex);

```

```

- root = cgrp->root;
- n = strlen(root->release_agent_path,
-   sizeof(root->release_agent_path));
- n = min(n, (size_t) PAGE_SIZE);
- strncpy(s, root->release_agent_path, n);
- mutex_unlock(&cgroup_mutex);
- s += n;
- break;
- }
- default:
-   retval = -EINVAL;
-   goto out;
- }
- *s++ = '\n';
-
-   retval = simple_read_from_buffer(buf, nbytes, ppos, page, s - page);
-out:
-   free_page((unsigned long)page);
-   return retval;
- }
-

```

```

static ssize_t cgroup_file_read(struct file *file, char __user *buf,
    size_t nbytes, loff_t *ppos)
{

```

```

@@ -2333,8 +2298,10 @@ static struct cftype files[] = {

```

```

static struct cftype cft_release_agent = {
    .name = "release_agent",
-   .read = cgroup_common_file_read,
-   .write = cgroup_common_file_write,
+   .read_seq_string = cgroup_release_agent_show,
+   .write_string = cgroup_release_agent_write,
+   .lockmode = CFT_LOCK_CGL,
+   .max_write_len = PATH_MAX,
    .private = FILE_RELEASE_AGENT,
};

```

```

@@ -3161,27 +3128,25 @@ static void cgroup_release_agent(struct
    while (!list_empty(&release_list)) {
        char *argv[3], *envp[3];
        int i;
-   char *pathbuf;
+   char *pathbuf = NULL, *agentbuf = NULL;
        struct cgroup *cgrp = list_entry(release_list.next,
            struct cgroup,
            release_list);
        list_del_init(&cgrp->release_list);
        spin_unlock(&release_list_lock);

```

```

    pathbuf = kmalloc(PAGE_SIZE, GFP_KERNEL);
-   if (!pathbuf) {
-       spin_lock(&release_list_lock);
-       continue;
-   }
-
-   if (cgroup_path(cgrp, pathbuf, PAGE_SIZE) < 0) {
-       kfree(pathbuf);
-       spin_lock(&release_list_lock);
-       continue;
-   }
+   if (!pathbuf)
+       goto continue_free;
+   if (cgroup_path(cgrp, pathbuf, PAGE_SIZE) < 0)
+       goto continue_free;
+   agentbuf = kmalloc(PATH_MAX, GFP_KERNEL);
+   if (!agentbuf)
+       goto continue_free;
+   strcpy(agentbuf, cgrp->root->release_agent_path);

    i = 0;
-   argv[i++] = cgrp->root->release_agent_path;
-   argv[i++] = (char *)pathbuf;
+   argv[i++] = agentbuf;
+   argv[i++] = pathbuf;
    argv[i] = NULL;

    i = 0;
@@ -3195,8 +3160,10 @@ static void cgroup_release_agent(struct
    * be a slow process */
    mutex_unlock(&cgroup_mutex);
    call_usermodehelper(argv[0], argv, envp, UMH_WAIT_EXEC);
-   kfree(pathbuf);
    mutex_lock(&cgroup_mutex);
+   continue_free:
+   kfree(pathbuf);
+   kfree(agentbuf);
    spin_lock(&release_list_lock);
}
spin_unlock(&release_list_lock);

--

```

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
